FAIRSECO: A Research Impact Portal for a FAIR Research Software Ecosystem

DR. SLINGER JANSEN
DR. SIAMAK FARSHIDI

This project plan is the starting point for students to 1) to express a preference for a project and 2) to use as information at the start of a project. It is also an annex to the contract.

1 Introduction

Reusing research software, the software that advances society by supporting and contributing to academic research, is hard. It is hard to find the algorithm needed, it is hard to extract only the source code that is needed, and it is hard to give credits to the original research software engineers that developed the software. We have developed a worldwide database of software that makes it easier to find out where software is reused. In this project we want to extend our software with an infrastructure that measures research software impact, with the goal of making software more FAIR, i.e., Findable, Accessible, Interoperable, and Reusable.

At UU we have developed a software search engine for the worldwide software ecosystem (SearchSECO). We have done so by mining software repositories on the method level, instead of the project or source-code level. Abstract representations of the methods are hashed and stored in a scalable distributed database that stores millions of methods from the most common programming languages. The database can be used to analyze software reuse at the method level, follow method evolution over time, and perform vulnerability detection across multiple projects.

The research software (ReSoft) ecosystem can benefit from our technology, as it enables, with a relatively small investment, research software engineers to rapidly find information about research software, that can improve FAIRness. Using a series of Github actions, we try to collect data about a research software project and store this on a portal that monitors research impact. In this way we make research software more Findable, Accessible, Interoperable, and Reusable (FAIR).

The intended beneficiaries of the project are Research Software Engineers (RSEs), who benefit from the FAIR Research Software Impact Portal (RSIP), which enables them to monitor the impact of their software and reward others for the software they reuse. Organizations that produce ReSoft can use the tools to report on the impact their software has made. Finally, empirical software engineering researchers, such as ourselves, can use the content in the SearchSECO DB to analyze the ReSoft ecosystem at a scale that was impossible before.

SearchSECO is an initiative that is currently supported by TU Delft, Utrecht University, CWI, and Leiden University from the academic perspective. SecureSECO is also supported by the Software Improvement Group, Centric, Cisco, and KPN Security. Finally, we have endorsements from the Secure Software Alliance, the Blockchain Coalition, and the Lisk Center Utrecht.

The SearchSECO initiative is made up of a team of academics, who have extensive experience in creating repository mining and parsing solutions. We have committed to this initiative for at least the next five years. The team is, amongst others, responsible for the parsing tool sets Rascal [5] (https://www.rascal-mpl.org/) and for the GHTorrent [14] (https://ghtorrent.org) archive, an archive of Github that has been collected through distributed clients. The team at Utrecht University is the main responsible for this project proposal and has experience in developing several academic tools, such as a blockchain selection tool (https://dss-mcdm.com/) and the SecureSECO ledger itself (https://SecureSECO.org/ledger).

The Software Project team will primarily work together with dr. Slinger Jansen as the customer. For validation purposes, we will meet with some of the other stakeholders, particularly those at the eScience Center. A visit to the eScience Center by the team is recommended.

2 Goal of the project

Measuring and showing research software impact is a complex task. In this project we try to support Research Software Engineers (RSEs) with tools and methods to provide better insight into the impact that their software makes. It is hard for creators of research software to show the impact that their code has made on the field, and only very coarse measures exist for evaluating the success of research software (ReSoft) [4, 6, 8, 18, 20, 31].

The Challenge: Creation of the FAIRSECO Software Platform for RSEs

We have created the SearchSECO software platform that efficiently finds methods in the worldwide SECO. This is a radical innovation in ReSoft engineering, as it enables high-performance worldwide code searches that cross-cut language ecosystems on a conceptually intuitive level for RSEs. Furthermore, with a relatively small investment, we unearth the relationships between code fragments, code files, and their projects on a worldwide scale for ReSoft, enabling a more robust method of measuring software impact [13]. Please note that the basic platform has been built and is currently distributedly gathering data. It consists of the SearchSECO DB containing all methods and their meta-data and a CLI for evaluating any project in a git repository against the SearchSECO DB.

SearchSECO has two main parts. First, a group of distributed clients automatically spiders and indexes ReSoft projects and automatically extracts and annotates software methods [17, KP2]. The annotations are domain specific keywords that are extracted from project descriptions, code comments, author data, and variable names [8]. Methods are stored and tagged with a unique hash made from their abstract syntax tree, making these software methods identifiable, even when cloned across different projects. An extra affordance of SearchSECO is that it becomes possible to identify projects that reuse methods from other ReSoft projects, providing a more fine-grained view on the impact that ReSoft has made.

- RQ1: How can method reuse be identified in the research software ecosystem?
- RQ2: How can research software impact be measured and made transparent?

We will use the SearchSECO infrastructure to create a specific Research Software category of projects and mine these. Using the data from the projects we can determine the impact of projects on society and the research software ecosystem.

2.1 What is the functionality of the software system?

2.1.1 The FAIR Research Software Impact Portal and Impact on Research-Software.nl

With the SearchSECO DB we easily identify the provenance of source code in different projects, something that is considered very challenging by others [21, 32]. The FAIR impact portal provides ReSoft projects with a web page and API that shows the data that has been collected about the project's reuse and impact.

The portal shows other projects where methods from the ReSoft project are reused. Secondly, the portal shows how often the software has been cited in articles, papers, and websites. Thirdly, the portal shows how often the project shows up in searches and provides details about how the project is structured through its call graph and about its overall quality. Finally, the portal can show the provenance of source code, i.e., which code in the project originates from other projects and whether the licenses from those projects do not conflict with this project. These data can easily be integrated into pages such as https://www.research-software.nl/graphs/.

Current Status: As SearchSECO is a running open source project under AGPL-3.0 license, its progress can be followed here: https://github.com/SecureSECO. The project status can also be followed on our Grafana: http://secureseco.science.uu.nl/ (u: user, pw: KgBbJ5bUsCbBBnW) where it can be seen that we have indexed approximately 13M unique methods (44M non-unique) and 87k Github projects on the 1st of April 2022. We currently parse Python, Java, C and variants, and Javascript.

We plan to identify research software on Github, for instance by looking for the citation.cff file in a repository. While not perfect, this is a starting point for identifying open source research software.

FAIRSECO enables research about:

- **Relationships between software projects.**
  - *Establish package dependencies and cohesion* - we analyze dependencies between packages and assess how strongly two components are coupled.
  - *Establish academic reuse* – We want to study how research software projects are related to each other and whether they sufficiently cite each other. In the long run we could even imagine a tool that warns authors that they have not sufficiently cited the software they reuse.
- **Identification of pseudocode algorithms** – We intend to study how pseudocode can be translated into methods in different languages and identified across the ecosystem. Using mutation generation, we hope to rapidly find these needles in the haystack. (Probably out of scope)

Please note that some of these are not directly attainable and that this project is merely a step towards achieving these ambitious research goals.

Please note that as the project is mostly about the meta-data of software projects, it is not needed to focus on one particular technology or language. However, we have to work with the restrictions of the technology we plan to use, such as FASTEN.

2.2 Deliverables

A set of RSE GitHub actions that RSEs can use to rapidly increase the FAIRness of their project. The information that is generated can be stored in the Github project information but should also be stored by default into the ReSoft impact portal. Some examples of the GitHub actions that we envision:

- **Howfairis** – Can be used to generate data about the current status of the FAIRness of ReSoft. For more information: https://github.com/fair-software/howfairis .
- **SearchSECO** – Do a check command and use the report to identify projects that reuse this project. A github action is already available. https://github.com/SecureSECO/SearchSECOController .
- **Citation.cff Generation** – See if it is possible to automatically generate a citation.cff file or update it as part of the GitHub action. https://citation-file-format.github.io/cff-initializer-javascript/#/ .
- **SBOM Generation** – See if it is possible to generate SBOMs out of the data that we have available.
- **Tortellini License Checking –** Potentially even with the use of the SearchSECO data: https://github.com/tortellini-tools/action .
- **TrustSECO measurement –** We can use TrustSECO to see if it is possible to show measurements of trustworthiness over time. https://github.com/SecureSECO/TrustSECO .

For reference, these are some of the data that have been suggested by the GitHub team working with the eScience Center. Some of these can be added to TrustSECO as part of this project, or be stored independently if only relevant for ReSoft:

- The software development practices are professional
- Follows fair software principles
  - Possible metrics like those used in howfairis
- Software Quality
  - Possible metrics provided by static analysis methods
- Makes efficient use of existing software
  - Possible metrics on dependencies
- Maintainability
  - Tests
    - Possible metrics on code coverage
  - Developer documentation
    - Possible metrics?
- The software is actively developed and maintained
  - Possible metrics: commit activity, release activity, number of contributors (bus factor)
- The Software adheres to Open Science principles
  - Possible metrics: licence, is it citable, archived, …
- The software is reproducible
  - Possible metrics: persistent identifiers, software deposited at long term archives, are containers provided
- The software is reusable
  - Proven reuse (by others)
    - Possible metrics: (manually) recorded reuse in projects
  - Discoverable
    - Possible metrics: is listed in registries
  - Trusted
    - Possible metrics: testimonials
- Transparency / welcomes critical review

  - Possible metrics: data on issue tracking, resolved issues
  - promotes accessibility
- The software is impactful
  - The software is used for research (cited in papers?)
    - Possible metrics: number of citations
- There is a user community
  - Possible metrics: number of downloads, number of users, quotes, testimonials, mentions in mainstream news
- Workshops and tutorials
  - Possible metrics: (manually) recorded outreach activities

While we do not expect all these data to be stored in our portal, some of these, such as citation counts, should be included in the RSIP.

Currently, SearchSECO is storing little meta-data about projects. One of the extensions needed is the **SearchSECO meta-data component**, which stores particular meta-data about a project, such as the read.me file. These meta-data are required to identify which projects contain usable methods, which projects depend on which others, etc.

A **dependency extraction mechanism** is needed, such as FASTEN [15], World-of-Code, or the data in libraries.io to extract the dependencies. We propose that we do not build these ourselves, but use common infrastructures such as the ones mentioned above, or extend them.

The most important front facing part of this project is the **reporting service for research software projects**. Basically, we need to take the output from SearchSECO and process it to generate a report about the scientific project reuse. We want to know which projects are dependent on this project, which of those are scientific, and which projects have copied code from this project or are copied into this project. The citation.cff file in Github can be used as a marker to identify whether a project is academic or not. As this project is starting later in the year, this project may be already finished by then.

Please note that the project is working towards a Minimum Viable Product of the infrastructure.

2.3 Application area

As the eScience Center is one of our most important partners, it is probably best to start with the software enlisted on their portal https://research-software.nl/. Towards the end of the project, we would like to propose changes to this site that incorporate parts of the FAIRSECO infrastructure.

2.4 Related work

Currently, there are several systems that will help us on our way. Looking at the system, we roughly identify the following components, and the associated related work:

- **Research-software.nl** already collects significant data about the research software ecosystem.
- **TrustSECO** – is another part of the SearchSECO project family. It collects meta-data about software projects, such as Github stars. We aim to reuse these data in the future.
- **The Parsers** – The team can use tools from the Rascal project for parsing code (https://www.rascal-mpl.org/) and in particular their M3 models. Furthermore, the team can

use tools from the FASTEN project at TU Delft (https://www.fasten-project.eu/). Finally, one of the most useful parsers the team can use is the Vuddy tool (https://github.com/iotcube/hmark), as it already hashes methods in code files [1]. Recently we also found SrcML (https://www.srcml.org/), which appears to be even more feature complete than Rascal for our goals.

While these tools provide some puzzle pieces, it is still possible that encounter challenging problems in the project. Depending on the time available we will decide which of these will be implemented in a somewhat agile fashion.

2.5 Boundary conditions

There are some boundary conditions:

- We must at least parse all projects of the Research-Software.nl infrastructure.
- The code needs to be published as open source on the SecureSECO GitHub (https://github.com/SecureSECO), ideally under the AGPL license.
- It is perfectly okay to add code to other open source projects (SearchSECO, TrustSECO, and others) if it is reported well in this project.

2.6 Why is the project interesting?

The project takes place in an exciting new domain, with exciting new trends. The FAIR movement is relatively new and is showing that software engineering is research too. Furthermore, research software engineering should be a possible career path for any computer scientist, as it doesn't necessarily require you to follow the path of a PhD student, yet you get to work with the most interesting types of problems in the world, related to performance problems, scalability problems, and brand new shiny technologies.

Another reason why this project is interesting is that there are currently around 15 people working on these initiatives, meaning that you will not be working completely isolated, but that there are people depending on your work. There is even some guarantee that your software will be used and worked upon in the future, so students are effectively working on their own legacy.

2.7 Why is the project interesting for students?

This project is developed in an academic context with very ambitious engineering goals. Being part of the SecureSECO project enables students to:

- Become part of an extensive team of researchers,
- Co-author an academic publication on SearchSECO,
- Collect data that might benefit their own research goals and future thesis projects, and
- Work with the latest technologies.

The idea is that students can put this project on their resume as it may garner significant visibility.

3 Use

3.1 User- or player perspective

Searching for Methods – See the description of Figure 1.

Project Level - One typical user scenario will be to take one project, for instance with a GitHub link, and analyze the full project. We will upload the project into our database and identify code clones in the code. The code clones will be reported back to the user, with the goal of providing insight into the code reuse that is happening in the project.

We hope in the long run that our infrastructure becomes part of the standard research software engineering process. RSEs can use the SearchSECO DB to scan their code regularly with the goal of ensuring that their code does not violate any licenses, does not include any vulnerable code, and is not using outdated code. We want to create a reference workflow for ReSoft engineering, which includes tools such as Tortellini (https://www.research-software.nl/software/tortellini-github-action) and SearchSECO in the workflow.

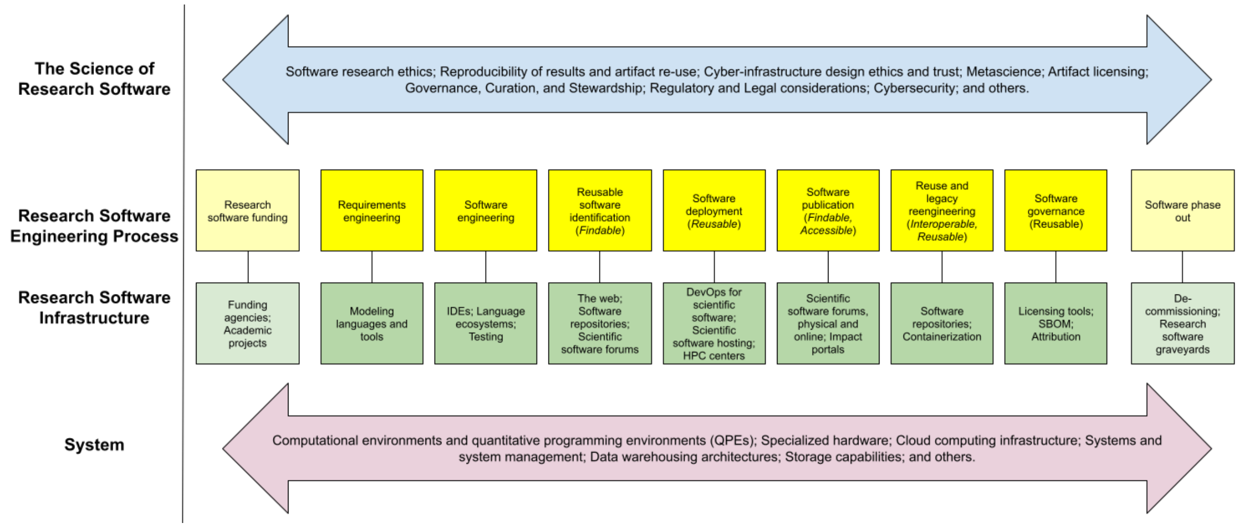
The Science of Research Software
Software research ethics; Reproducibility of results and artifact re-use; Cyber-infrastructure design ethics and trust; Metascience; Artifact licensing; Governance, Curation, and Stewardship; Regulatory and Legal considerations; Cybersecurity; and others.
Research Software Engineering Process
Research software funding
Requirements engineering
Software engineering
Reusable software identification (Findable)
Software deployment (Reusable)
Software publication (Findable, Accessible)
Reuse and legacy reengineering (Interoperable, Reusable)
Software governance (Reusable)
Software phase out
Research Software Infrastructure
Funding agencies; Academic projects
Modeling languages and tools
IDEs; Language ecosystems; Testing
The web; Software repositories; Scientific software forums
DevOps for scientific software; Scientific software hosting; HPC centers
Scientific software forums, physical and online; Impact portals
Software repositories; Containerization
Licensing tools; SBOM; Attribution
De-commissioning; Research software graveyards
System
Computational environments and quantitative programming environments (QPEs); Specialized hardware; Cloud computing infrastructure; Systems and system management; Data warehousing architectures; Storage capabilities; and others.

Figure 2: An example of the Research Software Development Workflow (loosely inspired by Stodden [4]). The SearchSECO project focuses on the FAIR parts of the process. In promoting and publishing the workflow, we will also highlight other tools such as Tortellini and AMUSE. This workflow also plays a large part in the first Workshop that we are planning.

3.2 System input

There are effectively two user scenarios. First, the system takes as input software projects from project repositories, which is already working. However, this needs to be extended so that the RSE can get a good overview of which projects use the project provided. Furthermore, we need to cache this data so that it can easily be reported in other data infrastructures.

3.3 System output

As output, a report is given on which code fragments are already in the database, under what license they are licensed, and in which projects they are present. We also provide a report on internal code clones in the project.

3.4 User behavior

A user may be an RSE trying to find out whether her software is being used by others in the field. She can do so by running her code against our infrastructure. This data can subsequently be reused in the Research-Software.nl directory, so the data must be stored somewhere.

Another user may be someone who is intending to reuse a ReSoft project. They can use the RSIP data to establish and evaluate the trustworthiness of the software.

3.5 Visualization and artwork

The project has a front page. As this is a scientific tool, there is no need for it to look exceedingly fancy, so no further design work is necessary.

4 Requisites

The tools that can be used for the project are typically open source and can easily be downloaded. The Primary Investigator is available as a customer on site. Currently, there are many students working on the SearchSECO project, so there is an active community that is easy to reach.

4.1 Domain knowledge and information supplied by the client

The PI will regularly (i.e., more than weekly) be available on site with the students.

4.2 Materials supplied by the client

We have a HPC SurfSARA account that can be used for software development. Furthermore, we aim to use a VPS from Utrecht University.

4.3 Data and examples supplied by the client

Most resources for this project are open source and openly available.

- The Research Software Directory at Research-software.nl.
- All github projects that have a citation.cff file.
- The SearchSECO DB.

4.4 Test environment

A portal is currently being developed where the SearchSECO infrastructure can be tried publicly. However, if you're interested in taking the platform for a spin, please take a closer look at https://github.com/SecureSECO/SearchSECOController.

5 Administrative

5.1 Contact person

Currently, the Primary Investigator is dr. Slinger Jansen at Utrecht University, who oversees the project. He is the main beneficiary and will actively stay in contact with the engineering team. However, depending on the part of the system, we can contact others in the consortium.

Furthermore, there are several PhD students working on this topic who need FAIRSECO as part of their research.

Finally, we have hired several student assistants who continue work on the SearchSECO and TrustSECO projects.

During the project, there will be one full-time PhD student available. This PhD student is working at the eScience center on this project. Furthermore, a student assistant who has worked on SearchSECO from the beginning is available for 20 hours per month if needed.

5.2 Project duration

We can have a project kick-off in the starting month. The project will last around 5 months. Around the beginning of final month, depending on project success, we can have a festive closing. We will present the system to the participating organization and in particular the eScience Center.

6 References

[KP1] S. Jansen, M. A. Cusumano, and S. Brinkkemper. **Software Ecosystems: Analyzing and Managing Business Networks in the Software Industry**. Edward Elgar Publishing, 344 pages, 2013.

[KP2] Slinger Jansen, Siamak Farshidi, Georgios Gousios, Joost Visser, Tijs van der Storm, Magiel Bruntink (2020) **SearchSECO: A Worldwide Index of the Open Source Software Ecosystem**. Proceedings of the 19th Belgium-Netherlands Software Evolution Workshop (pdf)

[KP3] Jansen, S. (2014) **Measuring the Health of Open Source Software Ecosystems: Moving Beyond the Project Scope** Accepted for publication in Information and Software Technology Journal, special issue on Software Ecosystems (pdf)

[1] A. Abadi, I. Ben-Harrush, N. Ifergan-Guy, and D. Pikus. Automatic extraction of sensitive code fragments to be executed in a sandbox, July 20 2017. US Patent App. 14/995,214.

[2] S. Bajracharya, T. Ngo, E. Linstead, Y. Dou, P. Rigor, P. Baldi, and C. Lopes. Sourcerer: a search engine for open source code supporting structure-based search. In Companion to the 21st ACM SIGPLAN symposium on Object-oriented programming systems, languages, and applications, pages 681–682, 2006.

[3] B. Basten, M. Hills, P. Klint, D. Landman, A. Shahi, M. J. Steindorfer, and J. J. Vinju. M3: A general model for code analytics in rascal. In 2015 IEEE 1st International Workshop on Software Analytics (SWAN), pages 25–28. IEEE, 2015.

[4] Stodden, V. (2020). The data science life cycle: a disciplined approach to advancing data science as a science. Communications of the ACM, 63(7), 58-66.

[6] D. Bollier and S. Helfrich. The wealth of the commons: A world beyond market and state. Levellers Press, 2014.

[7] S. Boshuis, T. B. Braam, A. P. Marchena, and S. Jansen. The effect of generic strategies on software ecosystem health: the case of cryptocurrency ecosystems. In Proceedings of the 1st International Workshop on Software Health, pages 10–17. ACM, 2018.

[8] M. L. Collard, M. J. Decker, and J. I. Maletic. srcml: An infrastructure for the exploration, analysis, and manipulation of source code: A tool demonstration. In 2013 IEEE International Conference on Software Maintenance, pages 516–519. IEEE, 2013.

[9] C. Erdmann, N. Simons, R. Otsuji, S. Labou, R. Johnson, G. Castelao, B. V. Boas, A.-L. Lamprecht, C. M. Ortiz, L. Garcia, et al. Top 10 fair data software things. 2019.

[11] M. Giancaspro. Is a 'smart contract' really a smart idea? insights from a legal perspective. Computer law & security review, 33(6):825–835, 2017.

[12] N. E. Gold, M. Harman, D. Binkley, and R. M. Hierons. Unifying program slicing and concept assignment for higher-level executable source code extraction. Software: Practice and Experience, 35(10):977–1006, 2005.

[13] G. Gousios, E. Kalliamvakou, and D. Spinellis. Measuring developer contribution from software repository data. In Proceedings of the 2008 international working conference on Mining software repositories, pages 129–132, 2008.

[14] G. Gousios and D. Spinellis. Ghtorrent: Github's data from a firehose. In *2012 9th IEEE Working Conference on Mining Software Repositories (MSR)*, pages 12–21. IEEE, 2012.

[15] Boldi, P., & Gousios, G. (2020). Fine-grained network analysis for modern software ecosystems. *ACM Transactions on Internet Technology (TOIT)*, *21*(1), 1-14.

[16] J. E. Hannay, C. MacLeod, J. Singer, H. P. Langtangen, D. Pfahl, and G. Wilson. How do scientists develop and use scientific software? In 2009 ICSE Workshop on Software Engineering for Computational Science and Engineering, pages 1–8. Ieee, 2009.

[17] D. R. Harris, A. S. Yeh, and H. B. Reubenstein. Extracting architectural features from source code. Automated Software Engineering, 3(1-2):109–138, 1996.

[18] W. Hasselbring, L. Carr, S. Hettrick, H. Packer, and T. Tiropanis. From fair research data toward fair and open research software. it - Information Technology, 62(1):39 – 47, 01 Feb. 2020.

[19] A. R. Hevner, S. T. March, J. Park, and S. Ram. Design science in information systems research. Management Information Systems Quarterly, 28(1):6, 2008.

[20] J. Howison and J. Bullard. Software in the scientific literature: Problems with seeing, finding, and using software mentioned in the biology literature. Journal of the Association for Information Science and Technology, 67(9):2137–2155, 2016.

[21] J. Howison, E. Deelman, M. J. McLennan, R. Ferreira da Silva, and J. D. Herbsleb. Understanding the scientific software ecosystem and its impact: Current and future measures. Research Evaluation, 24(4):454–470, 2015.

[22] Y.-Y. Hsieh, J.-P. Vergne, P. Anderson, K. Lakhani, and M. Reitzig. Bitcoin and the rise of decentralized autonomous organizations. Journal of Organization Design, 7(1):14, 2018.

[23] M. Hucka and M. J. Graham. Software search is not a science, even among scientists: A survey of how scientists and engineers find software. Journal of Systems and Software, 141:171–191, 2018.

[26] W. Jin, T. Liu, Q. Zheng, D. Cui, and Y. Cai. Functionality-oriented microservice extraction based on execution trace clustering. In 2018 IEEE International Conference on Web Services (ICWS), pages 211–218. IEEE, 2018.

[31] A.-L. Lamprecht, L. Garcia, M. Kuzak, C. Martinez, R. Arcila, E. Martin Del Pico, V. Dominguez Del Angel, S. van de Sandt, J. Ison, P. A. Martinez, et al. Towards fair principles for research software. Data Science, (Preprint):1–23, 2019.

[32] K. Li, P.-Y. Chen, and E. Yan. Challenges of measuring software impact through citations: An examination of the lme4 r package. Journal of Informetrics, 13(1):449–461, 2019.

[33] J. J. Marshall, S. Olding, R. E. Wolfe, and V. E. Delnore. Software reuse within the earth science community. 2006.

[34] J. Petke, M. Harman, W. B. Langdon, and W. Weimer. Specialising software for different downstream applications using genetic improvement and code transplantation. IEEE Transactions on Software Engineering, 44(6):574–594, 2017.

[37] R. Ramler, G. Buchgeher, C. Klammer, M. Pfeiffer, C. Salomon, H. Thaller, and L. Linsbauer. Benefits and drawbacks of representing and analyzing source code and software engineering artifacts with graph databases. In proceedings of the International Conference on Software Quality, pages 125–148. Springer, 2019.

[40] N. Sugawara and T. Yamamoto. Call graph dependency extraction by static source code analysis, Jan. 1 2013. US Patent 8,347,272.

[43] T. Zhang and M. Kim. Automated transplantation and differential testing for clones. In 2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE), pages 665–676. IEEE, 2017.

[44] Remaley, E. L., & Remaley, D. A. A. (2021). https://ntia.gov/. NTIA SBOM effort is progressing well.